Lab 4: Getting Started with ROS

Abstract

This lab is an introduction for ROS. We start with writing a simple “Hello World” program under the ROS environment. Next, we implement two nodes: publisher node and subscriber node to achieve message communications inside ROS. Then we use different tools that are provided inside ROS to help do some general debugging. The idea is to get familiar with the general ROS environment.

Objective

To learn the basic conceptual idea of ROS and gain practical knowledge of different operations inside ROS.

Theory

Robotic Operating System (ROS) is a platform for the development of robot software. It is an open source environment that allows users from all over the world to contribute toward the same or different projects in order to achieve their goal. One of its biggest advantages is the flexibility - the softwares that are used can be cross-platform without any limitations.

Nodes are the key element in ROS to perform communications between different hardware and software through a topic that the node either publishes or subscribes to. For example, a sensor node can publish its measurement to a particular topic and another control node subscribes to that topic in order to receive the sensor readings and make corresponding decisions for the robot.

The main programming language that is used in this lab is C++. But this is not the limitation for ROS, users can decide whatever programming languages that is needed for their application.

Procedures

Part 1: ROS Hello-world

First, we start by creating a “hello world” ROS node to learn the general procedure of setting up a project in ROS. For example, how to use different ROS commands, how to configure the CMakeList file, how to build the source files in ROS and run the executable, etc.

Part 2: Creating Publisher/Subscriber Nodes

Second, we create two different ROS nodes. One is the publisher node that constantly sends a message to a particular topic. The other one is the subscriber node that reads all the messages from the same topic and prints them onto the screen. This essentially is the basic idea behind ROS. Different publisher nodes can be established to read the data from sensors, cameras, encoders, etc. The subscriber nodes receives those data, processes the calculation, and eventually sends the control commands to the robot.

Part 3: Using ROS tools

There are plenty of ROS tools that can help developers to debug their implementation. For example, by using rqt_graph, you can examine the relationships between all the nodes and topics. Rqt_plot allows us to examine the scalar data and how it varies over the time. Furthermore, visualization tools such as Rviz, can simulate and plot different ROS messages.

Results

Figure 1 shows the demonstration of the "Hello World" ROS implementation.

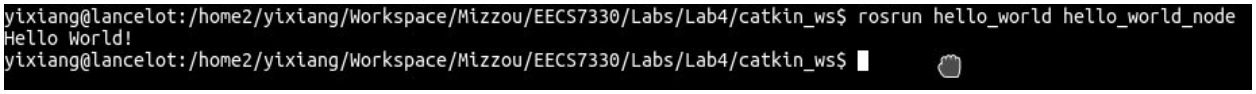

Figure 1. A ROS node that prints "Hello World" onto the screen.

The publisher and subscriber node are shown in Figure 2. In order for the subscriber to receive all messages, a slight delay is added inside the publisher node after its initialization.

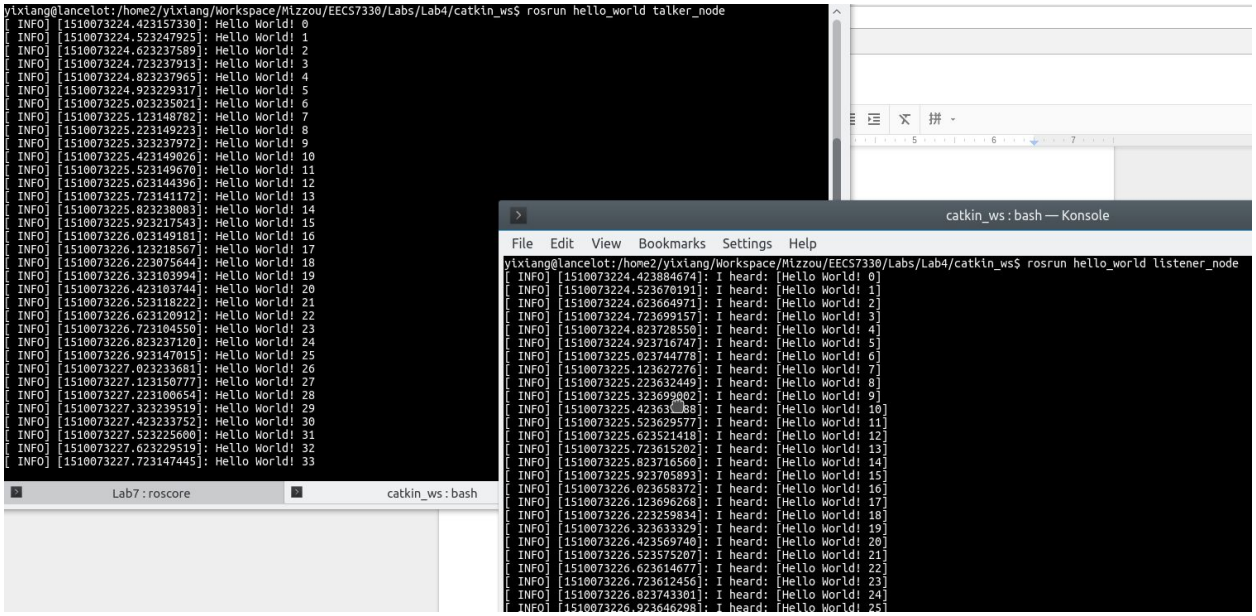

Figure 2. The publisher node and the subscriber node.

Figure 3 shows the ROS node graph between the publisher and the subscriber. The two nodes communicate with each other through the topic "/chatter".

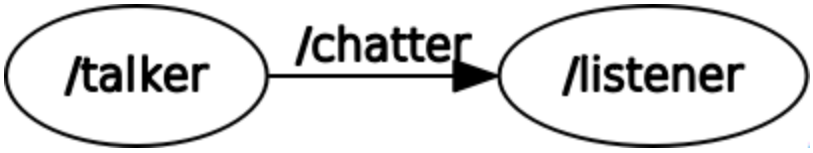

Figure 3. ROS node graph for the publisher and subscriber.

Conclusions

In this lab, we learn the fundamental ideal of ROS and some practical steps of how to create, configure, and operate under this robotics platform. By using two ROS nodes: a publisher and a subscriber, we gain more knowledge of ROS from a programming perspective as well as conceptual perspective for operating different robots. Finally, ROS also provides many different types of debugging tools that can help us in developing our projects. Learning how to use those tools are a necessary step as part of learning ROS.